AMENDMENTS IN THE CLAIMS

1. (currently amended) A system for periodically moving information units from a plurality of sources to an output destination based on information stored about each of the plurality of sources, the system comprising:

a time-based calendar which handles ~~some~~ scheduling of a first set of the information units based on the information stored about the plurality of sources;

a time-independent calendar which handles ~~others~~ scheduling of a remaining set of the information units that are not within the first set, said time-independent calendar handling the scheduling of the remaining set based on information stored about the plurality of sources and which places each source into a calendar location and ~~which~~ moves the source to a different place in the calendar of lower priority relative to a current calendar location of the source after servicing the source; and

a mechanism for determining when a flow is added to the source whether that source was at a first location in the time-based calendar and then (1) preventing the source from being placed at a second location that is ahead of a previously-calculated location in the time-based calendar and (2) placing the source at a ~~second~~ third location that is one of the previously-calculated location or a next location that is after the previously-calculated location within the time-based calendar, regardless of which position a time pointer of the time-based calendar is currently pointing to.

2. (currently amended) A method of servicing data flows placed into a queue for service in turn comprising:

calculating an initial new position in a calendar for a queue containing a serviced flow;

determining whether [[a]] the queue had a previous position in [[a]] the calendar;

if the queue had a previous position in the calendar, determining whether a new position ~~which~~ that would be assigned to [[it]] the queue is earlier than [[a]] the ~~previously-calculated~~ initial new position previously calculated in the calendar;

if the new position ~~which~~ that would be assigned is earlier than the previously-calculated initial new position, ~~using~~ assigning the previously-calculated initial new position to the queue;

and, if the previously calculated initial new position is not earlier than the new position ~~which~~ that would be assigned, ~~using~~ assigning the new position ~~which~~ that would be assigned.

3. (previously presented) A method including Claim 2 and further including considering the aging of the queue to determine whether stored parameters remain valid.

4. (canceled)

5. (previously presented) The system of claim 1 wherein the plurality of sources include a plurality of queues.

6. (previously presented) The system of claim 1 or claim 5 wherein the calculated location includes the location whereat the queue would have been attached upstream from the location whereat said queue was last serviced.

7. (previously presented) The method of claim 2 wherein using includes attaching the queue to the selected location.

8. (previously presented) The method of claim 6 wherein the stored information includes time stamps.

9. (currently amended) A system comprising:

a time-based calendar which handles ~~some~~ a first set of a plurality of information units based on the information stored about a plurality of sources; and

a mechanism for determining when a flow is added to the source whether that source was at a first location in the time-based calendar and then (1) preventing the source from being placed at a second location that is ahead of a ~~predefined~~ previously-calculated location in the time-based calendar and (2) placing the source at a ~~second~~ third location that is one of the previously-calculated location or a next location that is after the previously-calculated location within the time-based calendar, regardless of which position a time pointer of the time-based calendar is currently pointing to.

10. (currently amended) A method comprising:

providing at least one time based calendar having a plurality of locations and a time pointer moving relative to the plurality of locations as a result of scheduler ticks, each tick measured as a predetermined ratio of elapse time per pre-set number of bytes;

attaching a queue to a first calendar location whereat the time pointer is pointing;

servicing said queue by causing a frame to be transmitted from said queue whereupon said queue goes empty;

identifying a second location whereat the queue would have been re-attached had it not gone empty;

examining pre-defined characteristics associated with said queue to determine occupancy frames within said queue;

if examination indicates the queue is not empty, identifying a current location whereat the time pointer points;

correlating the current location of the time pointer and the second location; and

selecting a location which is not earlier than the second location to re-attach the queue, wherein when the current location of the time pointer is not earlier than the second location, the queue is reattached at the current location of the time pointer.

11. (previously presented) The method of claim 10 wherein the not emptied queue is attached to the selected location.

12. (previously presented) The method of claims 10 or 11 wherein the queue is attached by writing the i.d. (Identification number) of said queue in a stack located at each location.

13. (previously presented) The method of claim 12 wherein the stack is a Last In First Out (LIFO) stack.